STORIES

**D-Lib Magazine**
**September 1998**

ISSN 1082-9873

# Multilingual Federated Searching Across Heterogeneous Collections

James Powell
Distributed Information Systems, Virginia Tech
*Jpowell@vt.edu*

Edward A. Fox
Department of Computer Science, Virginia Tech
Blacksburg, Virginia 24061 USA
fox@vt.edu

## Abstract

This article describes a scalable system for searching heterogeneous multilingual collections on the World Wide Web. It details a markup language for describing the characteristics of a search engine and its interface, and a protocol for requesting word translations between languages.

## Table of Contents



## Problem

Search engines play a central role in helping users find information in digital libraries or on the Web. Complicating the situation, however, are the facts that:

- No search engine indexes all the content on the Internet

today [1].
- The various retrieval systems have different capabilities.
- Many digital libraries (for example, the Networked Digital Library of Theses and Dissertations [14]), have a distributed architecture (i.e., components and collections are located in separate places, coupled with a heterogeneous set of search engines).

Consequently, users are led to consult more than one search engine. While it would be helpful to automate this process, there are no widely accepted standards for submitting queries to multiple search engines simultaneously. Thus, there is a need for a simple, distributed retrieval system that can route searches to multiple search engines.

This requirement leads to a series of questions: Which search engines should be employed by a user to satisfy a given information need? What standards exist for describing search engines and their content [2], so they can be selected? Given the variations in capabilities among retrieval systems, how can an information need be mapped to those systems' respective query forms? Since most users perform only basic searches, how can suitable query forms be fed to a number of retrieval systems with minimal intervention by the users? At the same time, how can more complex queries and more sophisticated users be empowered to apply the full capabilities of more advanced search systems? Further, in a world where the Internet and digital libraries make available heterogeneous collections in scores of languages, how can multilingual searching become commonplace and easy? Finally, since this general problem seems complex, there is the question: How can the problem be subdivided into a series of manageable sub-problems?

One approach to metasearching -- that is, searching across search systems -- is the STARTS project [3]. The STARTS project [3] defines three main problems that need to be addressed by a metasearch system: selecting a source to query, evaluating the query at the source, and merging the results. In contrast, we considered the overall problem more broadly and subdivided it into nine specific problems:

1. determining the characteristics of a search site (search engine and collection),
2. selecting sites to search,
3. accepting and distributing a query,
4. overcoming language barriers,
5. mapping between search options,
6. dealing with results set variability (links, duplicates, etc.),
7. integrating results ranked in different ways,
8. handling response delays, and
9. distributing the workload.

The three STARTS problems correspond to our sub-problems 1 and 2; then 3, 4, and 5; then 6 and 7; respectively. Sub-problems 8 and 9 cover the added dimension of performance. Our solution, outlined below, addresses many of these problems, especially 1-5 and 9.

## Simple Search

Work on the problems listed above has proceeded for several decades [15]. Over the years, researchers have considered: vocabulary switching across databases, common command languages, resource discovery, source selection, de-duping, and various sort orderings of the search results. Some recent efforts, like the new MANTIS project at OCLC, employ new and powerful techniques for handling metadata, resource descriptions, distributed searching, and transforming results for flexible display [16]. Like MANTIS, many of the projects that seek to solve these problems require protocols, or developing protocols, with which search engines are or will be compliant. Of these, Z39.50 is the most popular and the most widely implemented. It supports resource discovery, query mapping, and results set merging. But it was originally designed for bibliographic systems containing highly structured collections of metadata. Most of the HTML content on the Web is anything but highly structured, so it is not surprising that document-oriented search engines do not support Z39.50.

Our approach to the problem was to start simple. To serve most users' needs, we focussed on simple queries. To avoid the work and negotiation involved in adding protocol support to diverse search systems, we created a new intermediate application that mediates search requests [3]. This solution masks the variations between search engine user interfaces and query interfaces. It also has the advantage of allowing additional manipulation of queries to improve retrieval effectiveness. This middle tier of software needs to have access to descriptions of the search engines' user interfaces, the types of queries supported, and the operators that define and qualify those queries. We envisioned that each site's description would include information about operators for formulating a query, and about user interface primitives (buttons, text fields, menus, etc.). We examined both Z39.50 and Stanford's front-end query language [4] to learn more about how to describe query operators and qualifiers in a platform-independent manner, and extended those ideas into the solution discussed below.

We arrived at that solution after noticing that one of the most common characteristics of web-based search engines is the so-called "simple search" option. Users simply enter a word, phrase, or set of words as a "natural language query" that is launched either by hitting the "return" key or clicking a button. We focussed on this single input field approach since most users submit short,

simple queries; appear confused about operators and other query enhancements [17]; and are usually forced to use separate forms or pages to submit more complex queries. The typical "simple search" became our guide for defining a query and user interface description language.

## Searchable Database Markup Language

We defined the Searchable Database Markup Language (SearchDB-ML), an application of the eXtensible Markup Language (XML), for describing a search site. We chose XML because it offers a great deal of flexibility compared to other alternatives such as storing site descriptions in a relational database. At the same time, XML supports our goal of having a human readable description that can be easily prepared by staff administering the federated search site by editing either a template or a copy of a related example.

SearchDB-ML uses a Document Type Definition "searchdb" for documents that describe retrieval resources. For simplicity, we concentrate on a "Lite" version of this language that only supports simple search interfaces and minimal query mapping. The full version of the language, which is still under development, will support search engines with both simple and complex interfaces. The example document below describes the simple interface to the Electronic Thesis and Dissertation collection at Virginia Tech.

Example:

```
<!DOCTYPE searchdb SYSTEM "searchdb.dtd">
<SEARCHDB>
    <ENGINE>
        <NAME>Opentext</NAME>
        <URL>http://www.opentext.com/</URL>
        <TYPE>Full text</TYPE>
    </ENGINE>
    <INSTANCE>
        <CONTENT>
            <TITLE>Virginia Tech Electronic Theses and Dissertations<
            <CREATOR>Scholarly Communications, Virginia Tech</CREATOR
            <SUBJECT>theses, dissertations</SUBJECT>
            <DESCRIPTION>Full text search of the archive of electroni
              and dissertations that have been approved by the Virgin
              Graduate School since 1995.
            </DESCRIPTION>
            <IDENTIFIER>http://scholar.lib.vt.edu/theses/etd-search.h
            <LANGUAGE TYPE="ISO6391988">en</LANGUAGE>
            <RIGHTS>Publicly accessible</RIGHTS>
        </CONTENT>

        <INTERFACE TYPE = "web" VERSION="simple">
            <URL>http://scholar.lib.vt.edu/otcgi/llscgi60.exe</URL>
            <FORMTYPE>post</FORMTYPE>
            <QUERYFIELD>
```

```
                    <FORMNAME>query</FORMNAME>
                    <MANDATORY/>
                </QUERYFIELD>
                <DATABASE>
                    <FORMNAME>db</FORMNAME>
                    <DEFAULT>0</DEFAULT>
                    <MANDATORY/>
                </DATABASE>
                <CONTROL>
                    <FORMNAME>mode</FORMNAME>
                    <DEFAULT>and</DEFAULT>
                    <OPTION>phrase</OPTION>
                    <OPTION>or</PHRASE>
                    <MANDATORY/>
                </CONTROL>
                <LANGUAGE TYPE="ISO6391988">en</LANGUAGE>
            </INTERFACE>

        <RESPONSE TYPE="html"/>

        </INSTANCE>
</SEARCHDB>
```

Each description contains two top-level document structures: <ENGINE> and <INSTANCE>. The <ENGINE> section is used to describe the actual search software employed by the site. It has three elements: <NAME>, <URL> and <TYPE>. This information is intended primarily to assist SearchDB-ML authors in identifying similar search engines to speed up the process of cataloging a new search site, since many descriptions are almost identical. It also is included to help users locate sites using the same search engine, whether for performance reasons or to allow them to take advantage of a particular search feature not available on other search engines. The <NAME> element should contain the name of the search software. <URL> should be the URL for the manufacturer of the software and <TYPE> can be any free text describing the basic characteristics of this particular engine.

The <INSTANCE> section describes a particular site using this software. It is further divided into <CONTENT>, <INTERFACE> and <RESPONSE> sections. The <CONTENT> section can contain any or all of the 15 elements described by the Dublin Core metadata standard [5]. These elements and their contents describe the type of collection, and can be used to determine if this site should be included in a federated search. The <INTERFACE> section describes one or more interfaces employed by the search engine. Finally the <RESPONSE> section describes the format of a response from this type of search engine. In Lite, the response only can be used to indicate the MIME document type of the results set.

The <CONTENT> section of the <INSTANCE> is primarily intended for resource discovery, which is becoming increasingly challenging. Publicly available web-based search engines already

number in the hundreds, and while there is a lot of overlap, there are also many specialized or foreign language sites that index and rank unique content not available through general purpose search engines like AltaVista. Fortunately, even the broadest collections indexed by search engines lend themselves to some level of classification. Every site must have a title, and it is almost always possible to determine who the author is (whether corporate or individual). Many search engines are accompanied by a portal site that classifies its collection into a hierarchy of top level categories that can be used as the contents of a subject tag. This same site gives a clue as to the primary language of the collection, which can be used both by the user and by the federated search software (for translation purposes). Other information also helps a user determine whether or not it is worthwhile to include an individual site in a query. Keywords are valuable, and usually are embedded in the search page. Understanding access restrictions can be critical; these are implicitly indicated by the accessibility of the site to other computers on the Internet.

Unlike the <CONTENT> section, the <INTERFACE> section is not intended for human users and contains little information that would be readily decipherable. Instead, this repeatable structure is used to describe an interface to a search engine. SearchDB-ML Lite is well suited to describing web-based search engines.

To prepare such a description, typically the first step is simply to map various HTML form elements to the appropriate SearchDB-ML elements, based on semantics. Later, then, for example, a user might select a particular database to search from a pull down menu, or indicate it by clicking a radio button next to an item in a list. For that user, the presentation is of no consequence, since the data delivered to the search engine looks the same in both instances. What is important in preparing a description is to indicate in the markup language that this data indicates which database the user wants to search, so it is described with the <DATABASE> element in SearchDB-ML.

Two other semantic elements are defined: <CONTROL> and <QUERYFIELD>. <CONTROL> is something of a general-purpose selector. It is primarily used to describe Boolean operators and other search modifiers, but it also can be used to describe results presentation options. The <QUERYFIELD> element is used exclusively to describe the field in which the user enters search terms. While the <CONTROL> and <DATABASE> elements typically have a default value defined using the <DEFAULT> element, <QUERYFIELD> elements rarely do. Instead, the Federated Searcher associates the translated and mapped query terms with this field before delivering them to each targeted search engine.

## Translation Request Protocol

We seek a simple solution, too, for the multilingual information retrieval (IR) problem. The increasing availability of public domain multilingual dictionaries on the Internet such as EDICT [6], CEDICT [7] and the Internet Dictionary Project [8] have made it possible to develop applications that implement simple dictionary-based translations. These can allow a searcher to overcome technical and linguistic barriers to information stored in languages not spoken or understood by the searcher. In particular, we focused on two cross-language retrieval tasks identified by Oard [19]:

- Search a monolingual collection in a language that the user cannot read.
- Retrieve information from a multilingual collection using a query in a single language.

Queries in one language are essentially useless for finding documents written or indexed in other languages. This is particularly true of queries formulated in Western languages that are targeting Asian language sites in China, Japan or Korea. Such queries are likely to produce no results (see Figure 1). However, if query terms are translated into the primary language of the collection, the likelihood of hits against the document collection increases [9].

Effectiveness, then, is largely limited by improper translations and by string matches that locate a term serving as part of an unrelated compound term (which is known as the "word division problem" [10]). For example, a query against Japanese search engines for "battery" might turn up a how-to guide for setting up a backyard pond, since the word for battery is composed of two characters: the characters for electricity and pond. Even searchers who understand the language of the target site often face these same problems (particularly in the case of Asian language search sites using Western search engines). While we recognize that "high precision is an important goal of a multilingual IR system" [9] and understand that there are more effective methods, we worked under the assumption that users would prefer some results rather than no results.

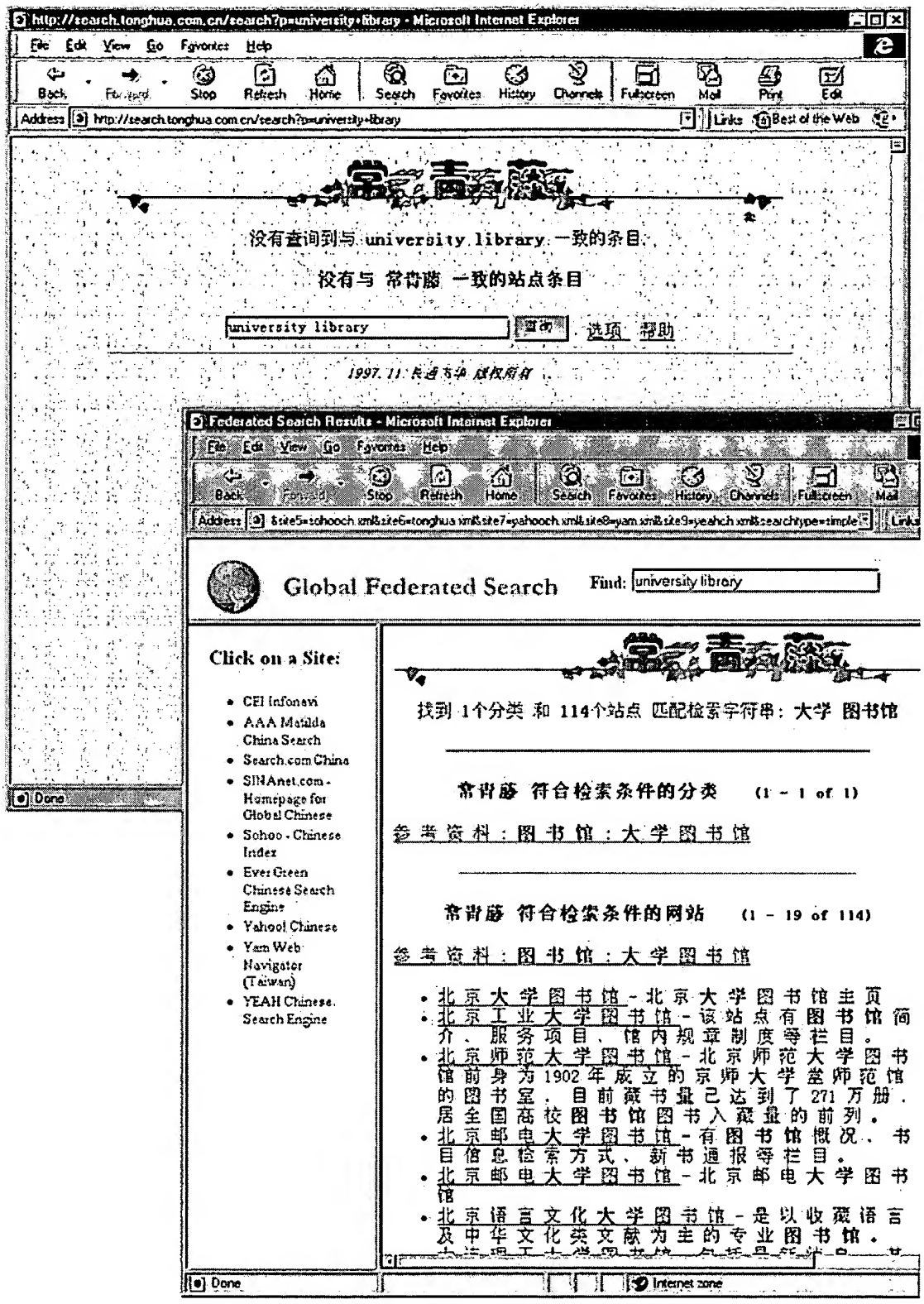


***Figure 1: English language query against Evergreen Chinese search engine at left yields no results; translated query submitted by Federated Searcher at right returns 114 hits.***

A technical problem faced by searchers who wish to bridge the language gap is character encoding and query term input. Many Asian languages use front-end processors that allow users to input

words phonetically using the Roman alphabet (transliteration). For example, many Japanese use the phonetic hiragana and katakana syllabaries when entering text, and then select the appropriate kanji characters from those discovered by the front end processor software during input. One of the common input techniques used in Chinese is a romanization system called pinyin. While technically similar to the Japanese input technique described above (e.g., it uses the Roman alphabet), it requires that the user have knowledge of the Mandarin dialect and written Chinese. So targeting both a Japanese search engine and a Chinese search engine would require not only mastery of both languages but also mastery of the input mechanism employed by software supporting each language, and the ability to recognize and correct errors introduced by transliteration [18]. Additionally, the software would have to understand and map between different popular character encoding schemes for each language. Finally, the user would have to have fonts for each language installed on their system to enable them to verify the correctness of a query. While such software is possible in a front-end processor, it is generally not included in Web browsers.

One solution to reading multilingual documents without installing fonts is the MHTML system developed by researchers at the University of Library and Information Science in Tsukuba, Ibaraki, Japan [20]. A Web gateway analyzes a requested multilingual HTML document, and then delivers the needed font glyphs in a resolution suitable for computer displays, along with the requested document. A Java-based MHTML browser uses this data to display the page within a regular Web browser. While this technology clearly points the way for future advances in networked multilingual software, it does not solve the multilingual input problem.

Query translation avoids all of these problems. It not only helps users to overcome the language barrier but also to surmount the associated technical barriers caused by inherently incompatible solutions to the problem of using a Western-style input device to author text in Eastern languages [11, 12].

Thus, a translation system that can both perform term translations and encoding conversion is desirable. To effect a modular solution that is simple and independent of development efforts for Web browsers, we introduce a translation request protocol (TRP) that supports language translation requests over the Internet. Any application that implements support for this protocol can request term translations, and even request that the results be delivered using a particular character encoding.

The core of the protocol is the TRANSLATE request, that must be supported by any application implementing TRP. It has five parts:

1. the TRANSLATE command,
2. the term to be translated,
3. the language of the term,
4. the target language, and
5. an optional character-set encoding identifier.

Example:

TRANSLATE library ENG JPN EUC

In this example, the request is for a translation of the word "library" from English to Japanese, to be provided in Extended Unix Code (EUC - a variable byte length encoding scheme for Japanese characters) character encoding, if possible.

Each part of the request is separated by one or more spaces, so spaces cannot occur in the term to be translated. This means the application must split a request into tokens and submit a translate request for each token. A three character Z39.53 [13] language code corresponding to the language desired (e.g., ENG, JPN) should be used to indicate the language of the term and the target language. The TRP protocol supports an optional NEGOTIATE command which can be used by the application to ask what languages are supported by a server. The character-set encoding identifier is a free text string field that indicates the encoding that the translation should be delivered in. Some valid character-set encoding identifiers include:

| Language | Character-set encoding identifier |
|---|---|
| Chinese | Big5 |
| English/Western | ISO-8859-1 |
| Japanese | EUC, JIS, Shift-JIS |
| Russian | ISO-8859-5 |
| Universal - Unicode | Unicode-2.0 |

Typically, the translations are stored in a database using a particular encoding, and those translations are delivered unmodified if encoding is not specified, or the specified encoding is not supported. Otherwise, the TRP server converts to the translation encoding before delivering the requested translation. Thus, by using TRP, an application can request word translations, and use those as appropriate. For example, it may reassemble a series of translated tokens for other uses, such as delivery to a remote search engine.

## Federated Searcher

The Federated Searcher is a Java-based server application that mediates user queries to multiple heterogeneous search engines. It consults a catalog of SearchDB-ML site descriptions to determine how the query should be formulated and how to deliver it to each search engine specified by the user. It uses a client package also written in Java to implement support for communications with a TRP server. When it determines the target site is not in English, it functions as a TRP client to request query term translations. It gathers results from each search engine and delivers an index of query results back to the user's Web browser. It also supports site discovery, query refinement and provides constant feedback about the status of each query. The prototype was tested with a catalog of over 20 sites including AltaVista, Bartlett's quotations and DejaNews.

## Experiences with NDLTD Implementation

One of the first production sites for the Federated Searcher was the Networked Digital Library of Theses and Dissertations [14] Federated Search (http://jin.dis.vt.edu/fedsearch/ndltd/support/search-catalog.html). NDLTD member institutions publish electronic versions of student theses and dissertations on the World Wide Web. While metadata about these publications typically makes its way to the library catalog of the university granting the degree, library OPAC's rarely provide support for full text searches. Since NDLTD focuses on students submitting their entire works, additional search systems supporting full text are utilized too. We built catalog descriptions of the search services provided by five institutions. Two sites use OpenText, one uses Dienst, another uses HyperWave while the fifth uses a Perl-based search script (search.pl). One of the five sites archives dissertations written in German. All were easily described with SearchDB-ML Lite. The Federated Searcher application was able to support cross-language retrieval, i.e., to submit queries and request translations for queries from English to the German site.

In the first two months of operation, over 300 NDLTD federated searches were logged. Some NDLTD users who have commented about the system find the site discovery step (Figure 2) somewhat cumbersome or

misunderstood its purpose (as can be seen from the log files). However, once they mastered this step they reported the Federated Searcher to be a useful tool for quickly distributing simple searches to multiple sites.

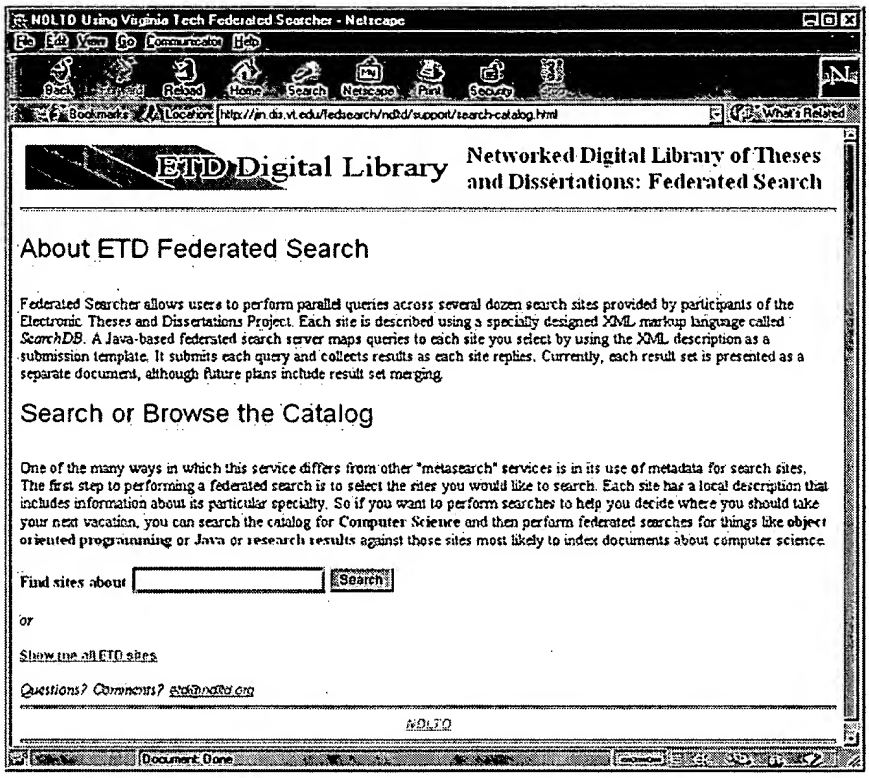


*Figure 2: NDLTD Federated Search site discovery (search catalog) page*

## Other Federated Search Collections

Two other collections of search engines have also been published with the Federated Searcher. The Global Federated Search site (http://jin.dis.vt.edu/fedsearch/) is the original prototype site which now allows users to perform federated searches against a collection of 92 search services from around the world including language specific sites in 7 different languages.

The Virginia Tech Federated Search (http://jin.dis.vt.edu/fedsearch/vt/support/search-catalog.html) is an "Intranet" search of various VT campus search services including the campus phone directory, Web search and more than a dozen departmental Web searches. These two sites should yield additional information about usage patterns and the types of queries people perform against multiple

sites.

## Conclusions and Future Plans

Of over 100 sites we have attempted to describe with SearchDB-ML Lite, only three had to be rejected. Two were eliminated because they maintained state with cookies, and the Federated Searcher implementation did not support this. A third Japanese language site could not be described because it utilized an intermediate query modification stage allowing the user to select one or more forms of their query using different Kanji or Kana representations.

The notion of a site description catalog was initially very appealing to some because it utilizes a simple markup language that can be quickly mastered by people familiar with HTML, which means that new sites can be added to the catalog of sites without programming. But upon closer review, some users have expressed concern that a short catalog description is not sufficient for describing a searchable archive of information and have suggested other mechanisms for collecting and constructing metadata about a search engine's content.

A review of the results generated from querying over 50 sites simultaneously reveals that in some cases more sophisticated query mapping is necessary to retrieve results sets that truly correspond to the original query. Some sites will apply a Boolean AND or OR to a multi-token string, while others treat it as a literal string. An extended version of the SearchDB markup language is being developed that can reflect the default and available query modifiers for each search engine; work is also underway to implement a mapping system that uses this information.

The Federated Search system begins to solve several problems facing Internet searchers:

1. By translating queries to other languages, it makes resources in other languages more accessible.
2. It allows searchers to locate and target the best search engines for their particular query.
3. It enables searchers to perform queries against search engines with which they are not familiar.

Some of these problems have been tackled before with

varying degrees of success. What differentiates the Federated Search system from previous solutions is the ease with which a new site can be added to a collection of search engines and its modular support for dynamic query translation. The only requirements for extending this system are a basic understanding of structured markup languages and some experience with web-based search engines. By defining an easy to understand structured markup language for describing Web search engines, we have made it possible for users to produce customized, flexible information retrieval tools that can be used to perform language- and search engine-independent queries on the global Internet and digital libraries.

## References

[1] Millions of Web pages overwhelm search engines, CNN Interactive, April 2, 1998.

[2] Sebastian Hammer and John Favaro, Z39.50 and the World Wide Web, D-Lib Magazine, March 1996,http://www.dlib.org/dlib/march96/briefings/03indexdata.html

[3] Luis Gravano, Kevin Chang, Hector Garcia-Molina, Carl Lagoze, Andreas Paepcke, STARTS: Stanford Protocol Proposal for Internet Retrieval and Search, http://www-db.stanford.edu/~gravano/starts.html

[4] Chen-Chuan Kevin Chang, Front-End Query Language,http://www-db.stanford.edu/~kevin/queryLanguage.html

[5] Stuart Weibel and Eric Miller, Dublin Core Metadata, http://purl.org/metadata/dublin_core

[6] Jim Breen, EDICT Project, http://www.dgs.monash.edu.au/~jwb/japanese.html#edict_proj

[7] Paul Denisowski, CEDICT (Chinese-English Dictionary) Project, http://www.mindspring.com/~paul_denisowski/cedict.html

[8] Tyler Chambers, Internet Dictionary Project, http://www.june29.com/IDP/

[9] David A. Hull and Gregory Grefenstette, Querying Across Languages: A Dictionary-Based Approach to Multilingual Information Retrieval, Readings in

Information Retrieval, Morgan Kaufmann, 484-492.

[10] Hsinchun Chen, An Automatic Indexing and Neural Network Approach to Concept Retrieval and of Multilingual (Chinese-English) Documents, http://ai.bpa.arizona.edu/papers/chinese93/chinese93.html

[11] Bradley D. Mohr, Guide to Japanese Computing, http://www.tjp.washington.edu/computing/japanese/guidetojapanesecomputing.

[12] Erik E. Peterson, Online Chinese Tools, http://www.erols.com/eepeter/chtools.html

[13] NISO Z39.53 Language Codes, http://www.oasis-open.org/cover/nisoLang3-1994.html

[14] Networked Digital Library of Theses and Dissertations, http://www.ndltd.org/

[15] Richard S. Marcus, Design Questions in the Development of Expert Systems for Retrieval Assistance, in Proceedings 49th Annual Meeting American Society for Information Science, Sep. 1986, Chicago, 185-189.

[16] Keith Shafer, MANTIS, OCLC Office of Research, Dublin, Ohio, 1998, http://orc.rsch.oclc.org:6464/

[17] Ghaleb Abdulla, Binzhang Liu, Rani Saad and Edward A. Fox, Characterizing World Wide Web Queries, Technical Report TR-97-04, 1997, Virginia Polytechnic Inst. and State University Dept. of Computer Science, available through http://www.ncstrl.org

[18] Christine L. Borgman, Multi-Media, Multi-Cultural, and Multi-Lingual Digital Libraries, D-Lib Magazine, June 1997, http://www.dlib.org/dlib/june97/06borgman.html

[19] Douglas W. Oard, Serving Users in Many Languages, D-Lib Magazine, December 1997, http://www.dlib.org/dlib/december97/oard/12oard.html

[20] Akira Maeda, Myriam Dartois, Takehisa Fujita, Tetsuo Sakaguchi, Shigeo Sugimoto, and Koichi Tabata, Viewing Multilingual Documents on Your Local Web Browser, Communications of the ACM, April 1998, 41(4): 64-65.

Copyright © 1998 James Powell and Edward A. Fox

---

Top | Magazine
Search | Author Index | Title Index | Monthly Issues
Previous Story | Next Story
Comments | E-mail the Editor

---

*hdl:cnri.dlib/september98-powell*